Advanced Java Programming (17625)

Event Handling

20 Marks

Specific Objectives

- To write event driven programs using the delegation event model.
- To write programs using adapter classes & the inner classes.

The Delegation Event Model

- Modern approach to handling events is based on the "*delegation event model*".
- Defines standard and consistent mechanisms to generate and process event's.
- "*Source*" generates an event and sends it to one or more *listeners*.

The Delegation Event Model : Event

- *Event* is an object that describes a state change in a source.
- Some action we have to performed when it is generated.
- Generated when user is interacted with components.
- Example: Pressing Button, Selecting item from list etc.
- It can be also generated when timer expires, counter value exceeds, software and hardware failure etc.

Event Source

- *Source* is an object that generates an event.
- This occurs when the internal state of that object changes.
- Sources may generate more than one type of event.
- Sources must be register listener so that listener will receive notifications.
- For Register/add:
  - public void add*Type*Listener(*Type*Listener *el*)
- For remove:
  - public void remove*Type*Listener(*Type*Listener *el*)

Event Listener

- *Listener* is an object that is notified when an event occurs.

- Two requirements:
  - It must have been registered with one or more sources to receive notifications.
  - It must implement methods to receive and process these notifications.

Event Classes

- Event Classes are core of Java’s event handling mechanism.
- **EventObject** is the root of the Java event class hierarchy which is present in **java.util**.
- It is the superclass for all events.
- Constructor: EventObject(Object *src*).
- EventObject class has defines two methods:
  - **Object getSource( ) :** method returns the source of the event.
  - **String toString( ) :** returns string equivalent of the event.

Event Classes : AWTEvent

- **AWTEvent** is defined in **java.awt** package.
- It is a subclass of **EventObject**.
- It is the superclass of all AWT-based events.

- **Summarize:**
- **EventObject** is a superclass of all events.
- **AWTEvent** is a superclass of all AWT events that are handled by the delegation event model.

Summary

- Event Source – the class which broadcasts the events
- Event Listeners – the classes which receive notifications of events
- Event Object – the class object which describes the event.

Event Classes : Diff classes

- ActionEvent
- ComponentEvent
- ContainerEvent
- FocusEvent
- ItemEvent
- KeyEvent
- MouseEvent
- TextEvent
- WindowEvent

ActionEvent

- Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
- **ActionEvent** class defines four integer constants that can be used to identify any modifiers associated with an action event:
  - **ALT_MASK**, (8)
  - **CTRL_MASK**, (2)
  - **META_MASK**, (4)
  - **SHIFT_MASK**. (1)
- In addition, an integer constant, **ACTION_PERFORMED (1001)**, which can be used to identify action events

ActionEvent

- Constructors:
  - ActionEvent(Object *src*, int *type*, String *cmd*)
  - ActionEvent(Object *src*, int *type*, String *cmd*, int *modifiers*)
  - ActionEvent(Object *src*, int *type*, String *cmd*, long *when*, int *modifiers*)
  - src: object which generate event
  - type: type of event
  - cmd: Command string
  - modifiers: which modifier key
  - when: when the event occurred

ActionEvent

- getActionCommand() used to get command name.
- int getModifiers() used to get modifier key.
- long getWhen( ) used to get when event generated.

ComponentEvent class

- A **ComponentEvent** is generated when the size, position, or visibility of a component is changed.
- There are four types of component events

  - COMPONENT_HIDDEN The component was hidden.
  - COMPONENT_MOVED The component was moved.
  - COMPONENT_RESIZED The component was resized.
  - COMPONENT_SHOWN The component became visible.

  - Constructor:
    - ComponentEvent(Component *src*, int *type*)

ContainerEvent class

- **ContainerEvent** is generated when a component is added to or removed from a container.
- Two Constants defined
  - **COMPONENT_ADDED** and
  - **COMPONENT_REMOVED**
- Subclass of ComponentEvent Class
- Constructor:
  - ContainerEvent(Component *src*, int *type*, Component *comp*)

FocusEvent class

- **FocusEvent** is generated when a component gains or loses input focus.
- Two constants defined:
  - **FOCUS_GAINED** and **FOCUS_LOST**.
- Constructors:
  - FocusEvent(Component *src*, int *type*)
  - FocusEvent(Component *src*, int *type*, boolean *temporaryFlag*)
  - Focus Event(Component *src*, int *type*, boolean *temporaryFlag*, Component *other*)
- **isTemporary( )** method indicates if this focus change is temporary.

ItemEvent Class

- **ItemEvent** is generated when a check box or a list item is clicked or when a checkable menu item is selected or deselected.
- Item events:
  - DESELECTED The user deselected an item.
  - SELECTED The user selected an item.
  - ITEM_STATE_CHANGED that signifies a change of state.
- Constructor:
  - ItemEvent(ItemSelectable *src*, int *type*, Object *entry*, int *state*)

KeyEvent Class

- **KeyEvent** is generated when keyboard input occurs.
- There are three types of key events:
  - **KEY_PRESSED**,
  - **KEY_RELEASED**, and
  - **KEY_TYPED**
- Constructor:
  - KeyEvent(Component *src*, int *type*, long *when*, int *modifiers*, int *code*)
  - KeyEvent(Component *src*, int *type*, long *when*, int *modifiers*, int *code*, char *ch*)

KeyEvent Class

- There are many other integer constants that are defined by **KeyEvent.**
- **VK_0** through **VK_9** and **VK_A** through **VK_Z** define the ASCII equivalents of the numbers and letters.
- VK_ENTER VK_ESCAPE VK_CANCEL VK_UP VK_DOWN VK_LEFT VK_RIGHT VK_PAGE_DOWN VK_PAGE_UP VK_SHIFT VK_ALT VK_CONTROL

MouseEvent Class

- Eight types of mouse events.
- The **MouseEvent** class defines the following integer constants
  - MOUSE_CLICKED The user clicked the mouse.
  - MOUSE_DRAGGED The user dragged the mouse.
  - MOUSE_ENTERED The mouse entered a component.
  - MOUSE_EXITED The mouse exited from a component.
  - MOUSE_MOVED The mouse moved.
  - MOUSE_PRESSED The mouse was pressed.
  - MOUSE_RELEASED The mouse was released.
  - MOUSE_WHEEL The mouse wheel was moved

MouseEvent Class

- **MouseEvent** is a subclass of **InputEvent.**
- **Constructor:**
  - MouseEvent(Component *src*, int *type*, long *when*, int *modifiers*, int *x*, int *y*, int *clicks*, boolean *triggersPopup*)

TextEvent Class

- These are generated by text fields and text areas when characters are entered by a user or program.
- **TextEvent** defines the integer constant **TEXT_VALUE_CHANGED**.
- Constructor:
  - TextEvent(Object *src*, int *type*)

WindowEvent Class

- There are ten types of window events.
- **WindowEvent** class defines integer constants:
  - WINDOW_ACTIVATED The window was activated.
  - WINDOW_CLOSED The window has been closed.
  - WINDOW_CLOSING The user requested that the window be closed.
  - WINDOW_DEACTIVATED The window was deactivated.
  - WINDOW_DEICONIFIED The window deiconified (min => Normal).
  - WINDOW_GAINED_FOCUS The window gained input focus.
  - WINDOW_ICONIFIED The window was iconified(Normal=>min)
  - WINDOW_LOST_FOCUS The window lost input focus.
  - WINDOW_OPENED The window was opened.
  - WINDOW_STATE_CHANGED The state of the window changed.

WindowEvent Class

- **WindowEvent** is a subclass of **ComponentEvent.**
  - WindowEvent(Window *src*, int *type*, Window *other*)
  - WindowEvent(Window *src*, int *type*, int *fromState*, int *toState*)
  - WindowEvent(Window *src*, int *type*, Window *other*, int *fromState*, int *toState*)

Adapter Class

- An adapter class provides an empty implementation of all methods in an event listener interface.
- Adapter classes are useful when you want to receive and process only some of the events that are handled by a particular event listener interface.
- Example:

Adapter Class : Different Classes

ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListener

Inner Class

- Inner class is class which defined in another class.
- In inner classes, the Adapter class will defined in same class.
- No need of passing reference of object as it is in same scope.
- Ex.

Anonymous Inner Class

- An *anonymous* inner class is one that is not assigned a name.
- Ex.

Event Listeners Interfaces

- Event Delegation Model has two parts: Sources and Listeners.
- When event generated, then event source invoked appropriate method defined by interface.

Action Listener Interface

- Defines one method to receive action events.
  - void actionPerformed(ActionEvent *ae*)

ComponentListener Interface

- Defines four methods to recognize when a component is hidden, moved, resized, or shown.
  - void componentResized(ComponentEvent *ce*)
  - void componentMoved(ComponentEvent *ce*)
  - void componentShown(ComponentEvent *ce*)
  - void componentHidden(ComponentEvent *ce*)

ContainerListener Interface

- Defines two methods to recognize when a component is added to or removed from a container.
  - void componentAdded(ContainerEvent *ce*)
  - void componentRemoved(ContainerEvent *ce*)

FocusListener Interface

- Defines two methods to recognize when a component gains or loses keyboard focus.
  - void focusGained(FocusEvent *fe*)
  - void focusLost(FocusEvent *fe*)

ItemListener Interface

- Defines one method to recognize when the state of an item changes.
  - void itemStateChanged(ItemEvent *ie*)

KeyListener Interface

- Defines three methods to recognize when a key is pressed, released, or typed.
  - void keyPressed(KeyEvent *ke*)
  - void keyReleased(KeyEvent *ke*)
  - void keyTyped(KeyEvent *ke*)

MouseListener Interface

- Defines five methods to recognize when the mouse is clicked, enters a component, exits a component, is pressed, or is released.
  - void mouseClicked(MouseEvent *me*)
  - void mouseEntered(MouseEvent *me*)
  - void mouseExited(MouseEvent *me*)
  - void mousePressed(MouseEvent *me*)
  - void mouseReleased(MouseEvent *me*)

MouseMotionListener Interface

- Defines two methods to recognize when the mouse is dragged or moved.
  - void mouseDragged(MouseEvent *me*)
  - void mouseMoved(MouseEvent *me*)

TextListener Interface

- Defines one method to recognize when a text value changes.
  - void textValueChanged(TextEvent *te*)

WindowFocusListener Interface

- Defines two methods to recognize when a window gains or loses input focus
  - void windowGainedFocus(WindowEvent *we*)
  - void windowLostFocus(WindowEvent *we*)

WindowListener Interface

- Defines seven methods to recognize:
  - void windowActivated(WindowEvent *we*)
  - void windowClosed(WindowEvent *we*)
  - void windowClosing(WindowEvent *we*)
  - void windowDeactivated(WindowEvent *we*)
  - void windowDeiconified(WindowEvent *we*)
  - void windowIconified(WindowEvent *we*)
  - void windowOpened(WindowEvent *we*)